1
Introdução

O desenvolvimento de software open source é uma prática comum na engenharia de software. Desenvolvedores se reunem vitualmente para desenvolver códigos para melhorar desempenhos e adicionar funcionalidades em códigos já existentes. De acordo com (HIPPEL, 2005) o desenvolvimento open source teve início na década de 60 enquanto cientistas e engenheiros acadêmicos trabalhavam em laboratórios e faziam contribuições uns nos projetos dos outros. Essas contribuições foram facilitadas e se tornaram mais comuns com o aumento da infraestrutura ao longo dos anos a medida que mais pessoas tinham acesso a computadores e internet. Com algumas decisões de negocios tomadas ao longo do tempo que retringias acessos a softwares e passou a torna-los comeciais o que levou respostas como a criação empresas como a Free Software Foundation, em que previa a colaboração em projetos abertos resolvendo questões de diretos autorais (HIPPEL, 2005). Hoje milhares de empresas possuem projetos open sources das quais também milhares de desenvolvedores do mundo todo participam. Os benefícios para o desenvolvedor participar de projetos open source são, entre outros, a construção de um portfolio, ganho de experiência em desenvolvimento, aprender novas tecnologias e linguagens de programação, poder ter no curriculo que participou de grandes projetos e conhecer pessoas do mundo todo.

Uma das plataformas mais difundidas entre os desenvolvedores para repositorio de código e colaboração é o GitHub. Nele, projetos open source são desenvolvidos e qualquer usuário do GitHub consegue acessar os códigos, pull requests e conversas trocadas em projetos open sources. Esse ambiente atrai o interesse da comunidade acadêmica, principalmente para a utilização em pesquisas relacionadas a engenharia de software. O GitHub através de sua API permite a extração de dados que podem ser usados como métricas para se descobrir e explicar fenômenos relacionados a equipes de desenvolvimento. Essas pesquisas possuem diversos benefícios para a indústria. Como por exemplo, o avanço de metodos para gerenciamento de projetos de tecnologia, minimizacao de problemas de qualidade de software como bugs e code smells.

Dentro do gerenciamento de projetos, o framework 5W2H hoje é amplamente usado em diversas metodologias como PMBOK e métodos ágeis. Mas ha relatos de que o 5W2H começou a ser usado no toyotismo ligado a gestão da qualidade para melhor a produção dos automoveis (OHNO, 2019). O framework se basea nas sete peguntas que deve sem respondidas para se ter um plano de ação de forma eficiente. As perguntas a serem respondidas são: WHAT? WHERE? WHO? WHEN? WHY? HOW? HOW MUCH? Para se completar as perguntas deve ser considerado a area que se está inserido (teecnologia, indústria, etc.). Hoje existem alguns estudos que relacionam os 2W2H com a engenharia de software (SALVADORI; MAGNAGO; DUTRA, 2021), (SANTOS; LUCENA, 2023), (SANTOS et al., 2023).

1.1 Descrição dos objetivos

Considerando esse contexto, o presente trabalho foi realizado com o objetivo de desenvolver um código que usando recursos dispiniveis na literatura, ajuda a classificar as perguntas feitas em PR de repositorios do GitHub para software abertos entre cada uma das sete perguntas basicas dos 5W2H e relacionar essas classificações com as métricas destes repositorios.

Para a metodologia completa foi necessário seguir os seguintes passos descritos na imagem:

Detalhamento sobre as etapas seguidas:

- 1) Extrair comentários de PR (Pull Resquest) através da API do GitHub - O presente projeto não contempla este item, já que para esse estudo os pesquisadores do grupo de trabalho já tinham esses dados extraídos da API do GitHub, usados em (BARBOSA et al., 2023);
- 2) Filtrar comentários (que sejám de humanos e que contenham perguntas) - Nessa etapa foram usadas perguntas documentadas nas três bibliografias encontradas que relacionam a engenharia de software com os 5W2H, essas literaturas são: (SALVADORI; MAGNAGO; DUTRA, 2021) , (SANTOS; LUCENA, 2023) e (SANTOS et al., 2023);
- 3) Buscar base de treinamento para a classificação das perguntas - Para essa etapa foi usada uma adaptação do algoritmo classificador de texto

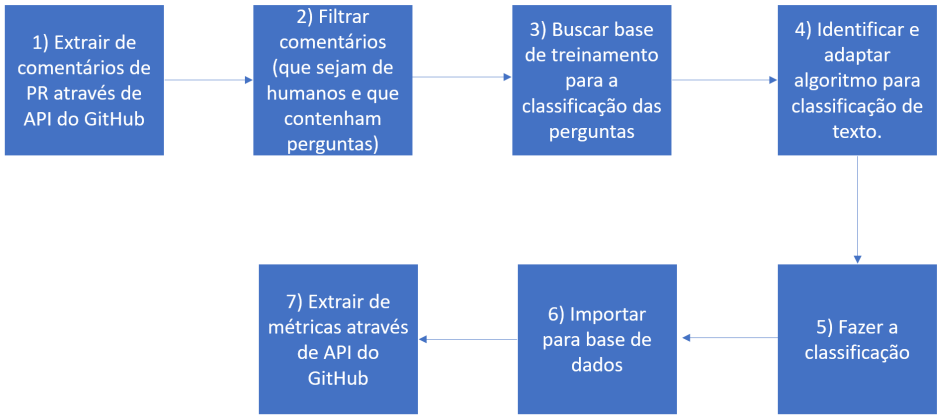

Figure 1.1: Etapas do projeto

nominado Gzip com KNN, desenvolvido por pesquisadores no seguinte artigo: (JIANG et al., 2022);

- 4) Identificar e adaptar algoritmo para classificação de texto;
- 5) Fazer a classificação - A classificação foi codificada de forma a apresentar a primeira opção de classificação, a segunda opção e suas respectivas probabilidades;
- 6) Importar para base de dados - os dados foram importados para o MONGOMongoDB;
- 7) Extrair de métricas através de API do GitHub - Da mesma forma que o item 1, esse trabalho também não possui o código dessa etapa pois esses dados já estavam disponíveis para a pesquisa.

2
Especificação de Requisitos

Com base no contexto e no objetivo apresentado foi possível definir os requisitos funcionais e não funcionais que estão detalhados a baixo.

2.1
Requisitos Funcionais

Considerando que os requisitos funcionais descrevem o que o sistema deve fazer, são funções deste sistema:

1. O sistema deve e fechar conexão com o banco de dados MongoDB;
2. Osistema deve selecionar os bancos de dados e coleções necessárias que estão armazenados no no MongoDB;
3. O sistema deve fazer a leitura do banco de dados;
4. O sistema deve filtrar os comentários, eleminando mensagens de bots e comentários que não possuem perguntas;
5. O sistema deve fazer treinamento considerando as perguntas encontradas na literatura (SALVADORI; MAGNAGO; DUTRA, 2021) , (SANTOS; LUCENA, 2023) e (SANTOS et al., 2023);
6. O sistema deve fazer a classificação dos dados usando o algoritmo Gzip com KNN disponível no trabalho (JIANG et al., 2022);
7. O sistema deve calcular as probabilidades associadas as classificações feitas no algoritmo Gzip com KNN.
8. O sistema deve permitir que o usuário manipule outros tipos de base de dados.

2.2
Requisitos não Funcionais

Já os requisitos não funcionais descrevem como o sistema deve ser para atingir o objetivo determinado. Neste trabalho desenvolvido tempos como requisitos não funcionais:

1. O sistema deve ser capaz de processar uma grande quantidade de dados em pouco tempo.
2. O sistema deve ser capaz de acessar apenas dados autorizados.
3. O sistema deve ser de fácil adaptação para manipular diferentes bases de dados.
4. O sistema deve permitir a implementação de novas funcionalidades.
5. O sistema deve ser confiável.

3
Atores

São identificados dois principais atores para o sistema:

1. Pesquisadores que precisam analisar informações de repositório do GitHub;
2. Gestores que querem acompanhar o desempenho da equipe e melhorar a comunicação entre a equipe de desenvolvimento.

4
Modelo de dados

O sistema em questão assume que os dados que serão analisados já foram extraídos e estão na disponíveis no MongoDB. Então para esse sistema é usado uma combinação de classes que estão no MongoDB com as que estão em python. Segue a seguir uma breve explicação de cada uma das classes:

4.0.1
A classe Comments

A classe commments já estava disponível no MomgoDB, ela é resultado de extração feitas do GitHub anteriormente a este trabalho por outros pesquisadores. A seguir estão detalhados os seus atributos.

- _id
- url
- html_url
- issueurl
- id
- user
- created_at
- updated_at
- author_association
- body
- reactions
- performed_via_github_app
- issue_number
- total_words

4.0.2
A classe SelectedData

A classe selectedData foi criada em python para extrair apenas as informações necessárias da classe comments. Ela possui os seguintes atributos:

- user - importante para identificar se a mensagem foi enviada por um robô ou por um humano.
- commentsData - é um comentário, na classe do mongoMongoDB, esse atrubuto se chamava "body"
- issueNumber - número do issue referente a esse comentário

No código essa classe teve que passar por alguns filtros. Primeiro, embora o campo "user" já tenha feito algumas classificações de quando se tratava de uma mensagem de robô, algumas mensagens estavam classificadas como humanos, mas eram de robô. Depois foi feito um filtro para de manter apenas as mensagens que tinham um "?", pois isso ajudou a filtrar todas as perguntas.

4.0.3
BDData

A classe BDData foi criada em também em python e salva o resuldado da análise feita pelo algoritmo usado para classificar as perguntas nos 5W2H.

- user - já filtrado para só conter mensagens de humanos
- commentsData - Aqui já estava filtrado para os comentários que possuem '?'.
- issueNumber - número do issue referente a esse comentário

4.0.4 Metrics

Assim como a classe Comments, a classe Metrics também já existia e estava disponível para essa pesquisa no MongoDB. Segue abaixo os atributos dela.

- _id
- issue_number
- discussion_duration
- discussion_size
- contributors
- core_developers
- newbies
- mean_number_of_words
- mean_time_between_comments
- number_of_comments
- density_design_keywords
- density_refactoring_keywords
- number_design_keywords
- number_refactoring_keywords
- class_degradation_density
- class_degradation_diversity
- class_design_change_density
- class_design_change_diversity
- class_diff
- diff_class_lvl
- diff_method_lvl
- method_degradation_density
- method_degradation_diversity

- method_design_density
- method_design_diversity
- method_diff
- num_class_lvl
- num_method_lvl
- number_females
- number_males

4.1 Diagrama de classe

A representação a seguir do diagrama de classe mostra que a classe Comments possui multiplicidade multipla em relação a classe SelectedData. Isso acontece devido aos filtros aplicados que foram explicados anteriormente. Já a classe SelectedData possui multiplicidade 1 para 1 da classe BDData. Cada registro na classe BDData significa um diferente comentário que possui "?" no texto. Cada issue pode conter diversas perguntas, ou seja, diversos "?" e é por isso que a multiplicidade do BDData para o Metrics é muitos para 1, já que a classe Metrics possui um registro por issue.

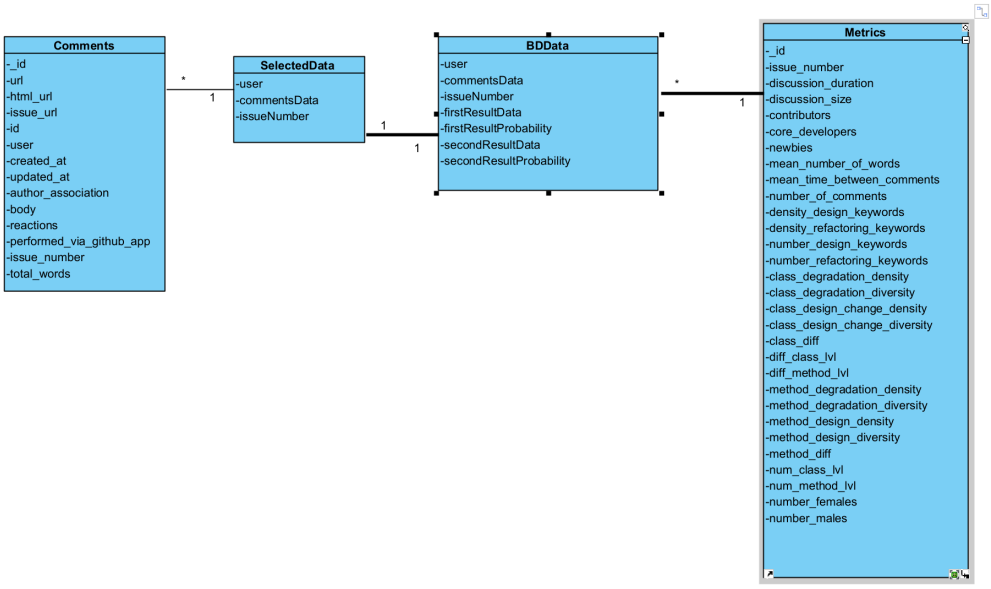

Figure 4.1: Diagrama de classe do sistema

5
Especificação Técnica e descrição funcional do software

O sistema foi feito na linguagem python e faz uso das seguintes bibliotecas:

- **gzip:** usada para trabalhar com arquivos.
- **json:** usada para trabalhar com dados no formato JSON.
- **pymongo:** usada para estabelecer conexão com o MongoDB e poder exportar, importar e atualizar os dados que estão ali salvos.
- **numpy:** Fornece suporte para matrizes multidimensionais

O sistema se comunica com o usuário através do console e fazendo extrações e salvando dados no MongoDB. O código possui 3 arquivos: gzip_classification.py, examples.json e resultsAnalysis.py. No arquivo gzip_classification.py são declaradas duas classes, a SelectedData e BDData, como são mostradas nas imagens a seguir:

```
#creating a class for the the data extraction
class SelectedData:
    def __init__ (self, user, commentsData, issueNumber):
        self.user = user
        self.commentsData = commentsData
        self.issueNumber = issueNumber
```

Figure 5.1: Classe SelectedData

A classe SelectedData armazena os dados extraídos do MongoDB, enquanto a classe BDData filta os dados que serão analizados e já possui espaço para armazenar os resultados das análises.

Os dados para teste estão no arquivo Examples.json, o conteúdo desse arquivo são as frases que revisão de literatura classifica como WHO, WHERE, WHAT, WHEN WHY, HOW ou HOW MUCH. Como mostram as imagens a seguir.

```python
#creating a class for the bd data
class BDData:
    def __init__ (self, user, commentsData, issueNumber):
        self.user = user
        self.commentsData = commentsData
        self.issueNumber = issueNumber
        self.firstResultData = ""
        self.firstResultProbability = 0
        self.secondResultData = ""
        self.secondResultProbability = 0

    def resultK1(self, firstResultData, firstResultProbability):
        self.firstResultData = firstResultData
        self.firstResultProbability = firstResultProbability

    def resultK2(self, secondResultData, secondResultProbability):
        self.secondResultData = secondResultData
        self.secondResultProbability = secondResultProbability

    def printResults (self):
        print(f"\nUser: {self.user}")
        print(f"\nComments: {self.commentsData}")
        print(f"\nIssue Number: {self.issueNumber}")
        print(f"\nfirst Result Data: {self.firstResultData}")
        print(f"\nprob first Result Data: {self.firstResultProbability}")
        print(f"\nsecond Result Data: {self.secondResultData}")
        print(f"\nprob second Result Data: {self.secondResultProbability}")
        print(f"\n\n********************\n\n")
```

Figure 5.2: Classe BDData

A seguir estão todas as frases usadas no arquivo de treino separadas por literatura. Foram no total vinte e duas frases de Santos e Lucena Santos e Lucena (2023), vinte e oito frases de Salvadoreri, Magnago e Dutra Salvadori, Magnago e Dutra (2021) e vinte e nove frases de antos et al. Santos et al. (2023). No total foram levantadas para treinamento treze frases da categoria "WHAT", nove frases da categoria "WHY", onze frases da categoria "WHEN", dez frases da categoria "WHERE", treze frases da categoria "WHO", dezenove frases da categoria "HOW" e quatro frases da categoria "HOW MUCH".

Após fazer todos os filtros necessários nas mesnsagens e também fazer o treinamento com as frases da revisão de literatura, foi usada uma adaptação do algoritmo desenvolvido por Jiang et al Jiang et al. (2022), o código está na figura

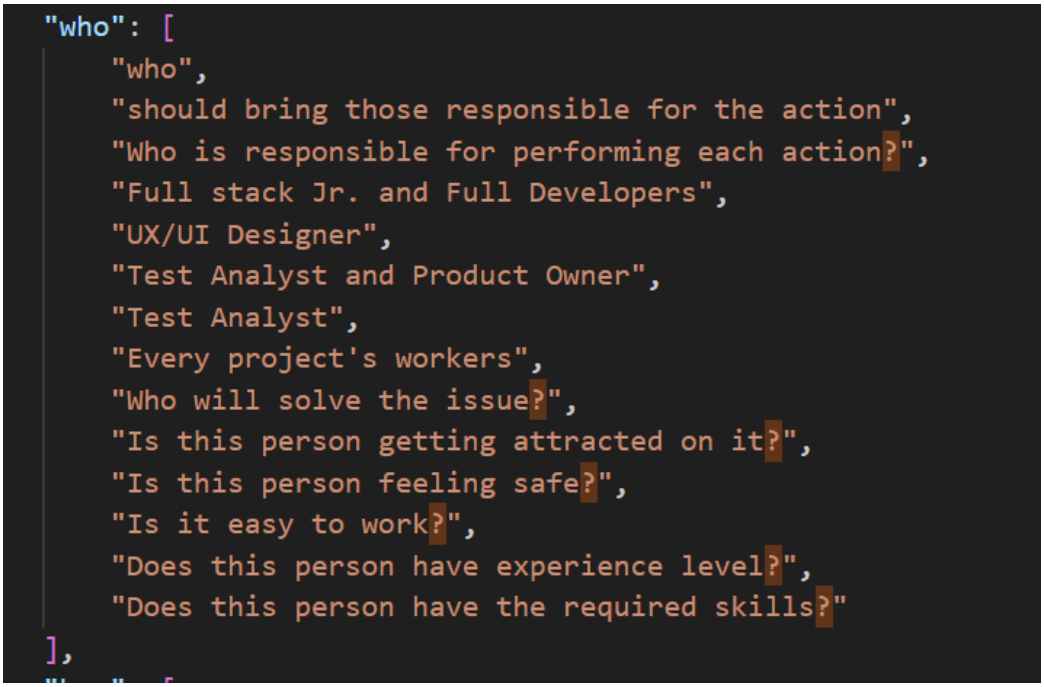

Figure 5.3: Frases usadas para treino da classificação "WHO" escritas no arquivo Examples.json

a seguir.

Após a classificação ser feita, ela é salva em uma variável da classe BDData e os dados são enviados novamente para o MongoDB.

What	Why	When	Where	Who	How	How Much
"brought the actions, the steps"	"brought justifications and reasons"	"brought dates, deadlines and periodicities"	"brought location within the system architecture"	"should bring those responsible for the action"	"presented methods and processes"	"brought cost data to the solution"
"What is considered when making the decision?"	-	"What is the frequency of carrying out this action?"	"Where does the current system run?"	"Who is responsible for performing each action?"	"How does the Negotiation Process work?"	
"What are the main difficulties in decision-making?"	-				"How to improve the decision-making process?"	
"What are the main logistical problems of this process?"	-				"How to streamline the decision-making process?"	
"What is the scope of this process?"					"How is currently existing decision-making classified, programmed or non-programmed (or both)?"	
"What level of decision-making should the system handle? 5.1- Strategic Decision; 5.2- Tactical Decision; 5.3-. Operational Decision"					"How would you rate the decision according to probability? 5.1- Risk decision? 5.2- Decision uncertainty? 5.3- Sure decision?"	
					"How would this decision be classified according to the deadline? 6.1- Short-Term Decision? 6.2- Long-Term Decision?"	
					"How can we improve the current form?"	
Fonte: Santos, G. N. P., & De Lucena, C. J. P. (2023). Agile meeting method for building intelligent decision support systems. MethodsX, 11, 102311.						

Figure 5.4: Frases de Santos e Lucena Santos e Lucena (2023) usadades para treino

What	Why	When	Where	Who	How	How Much
"Automated testing before code and code review."	"The developer will have a revised code delivery through automated testing."	"During coding, the Developer uses the FDD before starting the task and after coding, completing the automated test."	" In localhost and code development platform."	"Full stack Jr. and Full Developers"	"Automated testing training for Full stack Junior Developers."	"What is the cost involved in carrying out the process?"
"Inclusion of the Design Review column to check each component of the screen delivered by the Developer, check font size and compare parameters and components with the prototype."	"The Designer will be able to do the revision and point out some faults before the test, making it less overloaded and preventing more bugs from opening."	"This column will be included after each task delivered by the developer and before the quality test. After that, the approved screen will be sent to the test analyst."	"Click up system in the current Sprint development - management and activity control tool. 3. Click up in the Bug area - activity management and control tool."	"UX/UI Designer"	"Including as a new task for the Project Designer."	
"Time metrics for the correction of each bug which must start in accordance with the degree of criticality, not by the opening date."	"Greater control of the completion time of each bug, to decrease their accumulation at the end of Sprint."	"When a bug is opened, the test analyst should communicate the PO who will determine the degree of priority and estimate the time of completion"	"In the test approval area."	"Test Analyst and Product Owner"	"Assigning as a new process of control and management of the opening and correction of bugs."	
"Holistic revision of the components of the prototype compared to the developed screen."	"To have a more reliable check of the components that were previously unnoticed, decreasing bugs in the client."	"At the time of manual testing."	"Along the Sprint and management by the Click Up tool."	"Test Analyst"	"Including as a manual testing activity."	
		"At Scrum cerimonies and Sprint development."		"Every project´s workers"	"Including as a second agile tool, through the verification of macro features and their user stories and inclusion of rules and comments in the code."	
Salvadori, B. G., Magnago, P. F., & Dutra, A. C. (2021). Project based on Agile Methodologies by DMAIC. In ICEIS (2) (pp. 337-344).						

Figure 5.5: Frases de Salvadoreri, Magnago e Dutra Salvadori, Magnago e Dutra (2021) usadas para treino

What	Why	When	Where	Who	How	How Much
Whats is the issue?	Why it is an issue?	When the issue was or will be solved?	Where is the issue?	Who will solve the issue?	How to solve the issue?	How big is the issue?
What is the issue's description?	What is the goal to solve it?	What is the deadline?	What is the start point?	Is this person getting attracted on it?	What are the steps to reproduce it?	What are the required effort?
What is the typo of the issue?	What are the benefits of solving it?	Is it open, closed or ongoing?	What are the connected areas?	Is this person feeling safe?	What are the steps to debug?	
	What are the expected behaviour?	What is the description of the ongoing solution?	Where is it located in code?	Is it easy to work?	What is the scenario?	
				Does this person have experience level?	What are de previous attempts?	
				Does this person have the required skills?	What are the solving challenges?	
Santos, F., Vargovich, J., Trinkenreich, B., Santos, I., Penney, J., Britto, R., ... & Gerosa, M. A. (2023). Tag that issue: Applying API-domain labels in issue tracking systems. arXiv preprint arXiv:2304.02877.						

Figure 5.6: Frases de Santos et al. Santos et al. (2023) usadas para treino

```python
def gzip_text_classification(training_set, test_set, k):
    training_set = np.array(training_set)
    test_set = np.array(test_set)
    pred_test_set = []

    for (x1, _) in test_set:
        Cx1 = len(gzip.compress(x1.encode()))
        distance_from_x1 = []
        for (x2, _) in training_set:
            Cx2 = len(gzip.compress(x2.encode()))
            x1x2 = " ".join([x1, x2])
            Cx1x2 = len(gzip.compress(x1x2.encode()))
            ncd = (Cx1x2 - min(Cx1, Cx2)) / max(Cx1, Cx2)
            distance_from_x1.append(ncd)

        sorted_idx = np.argsort(np.array(distance_from_x1))
        top_k_class = list(training_set[sorted_idx[:k], 1])

        unique_values, counts = np.unique(top_k_class, return_counts=True)
        relative_frequencies = counts / len(top_k_class)

        soft_prediction = {classname: 0 for classname in unique_values}
        for i, classname in enumerate(unique_values):
            soft_prediction[classname] = relative_frequencies[i]

        sorted_classes = sorted(soft_prediction.items(), key=lambda item: item[1], reverse=True)
        first_class, first_prob = sorted_classes[0]
        second_class, second_prob = sorted_classes[1] if len(sorted_classes) > 1 else (None, 0)

        pred_test_set.append((x1, first_class, first_prob, second_class, second_prob))
    return pred_test_set
```

Figure 5.7: Algoritmo adaptado de Jiang et al Jiang et al. (2022)

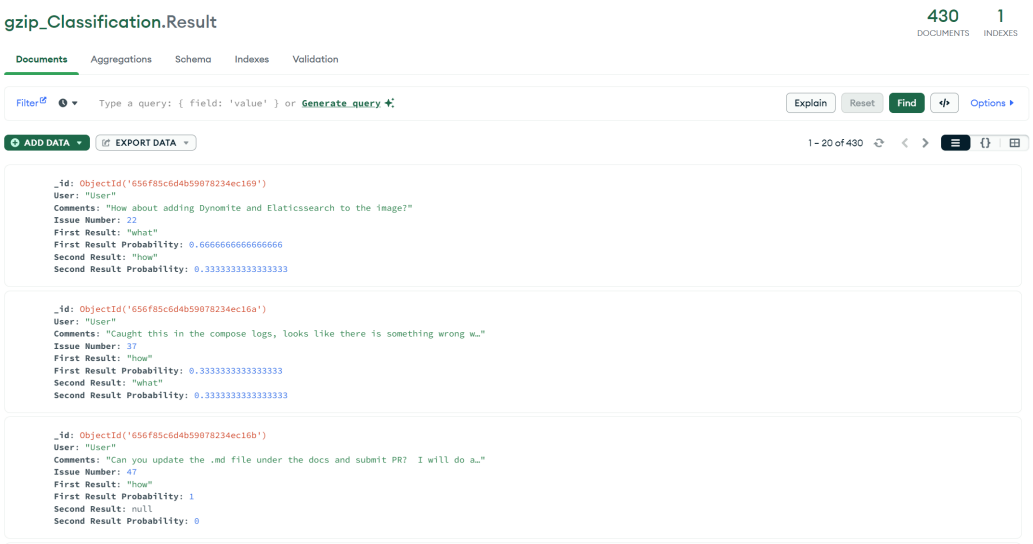

Figure 5.8: Screenshot dos Resultados gerados pelo sistema no MongoDB

6
Manual de Ultilização do sistema

Nos subtopicos a seguir será apresentado o Manual de utilização do sistema.

6.1
Introdução

O sistema possui uma proposta de auxiliar pesquisadores e gestores na análise de conversas e métricas de Pull Requests em repositorios do GitHub. Esta é uma versão inicial do sistema, que será melhorado a durante o proximo semestre e poderá recer incrementos durante sua vida util. O sistema ainda não possui uma interface de interação com o usuário, sendo necessário fazer alterações no código caso o usuário queira fazer consulta em uma base de dados que seja difente. O sistema está em desenvolvimento para auxiliar o trabalho de pesquisa que está em andamento pela autora e seu orientador.

6.2
Funcionalidades

São funcionalidades desse software:

- Extrair dados do MongoDB;
- Filtar dados pertinentes a análise;
- Fazer treinamento com os dados ja classificados anteriormente;
- Fazer a classificação dos dados obtidos nos repositorios do GitHub;
- salvar os dados no MongoDB.

6.3
Usuários

O sistema possui dois potenciais usuários, 1) pesquisadores de engenharia de software que possuem interesse em pesquisar conteúdos dos repositorios do GitHub e, 2) Gestores que possuem como objetivo melhorar o desempenho de suas equipes.

6.4
Benefícios do uso do Sistema

Entre os benefícios esperados pelo sistema estão: Tornar mais acessivel dados os dados para serem analisados, criar uma nova forma de análise da gestão de desenvolvedores.

6.5
Acesso ao Sistema

O acesso ao sistema vai acontecer através do repositorio do GitHub.

6.6
Utilização do Sistema

Nesta primeira versão do sistema, o usuário precisa fazer algumas alterações no código para atender as suas necessidades, conforme descrito a seguir:

1) No arquivo Example.json é necessário incluir os exemplos a serem treinados conforme as necessidades do usuário.

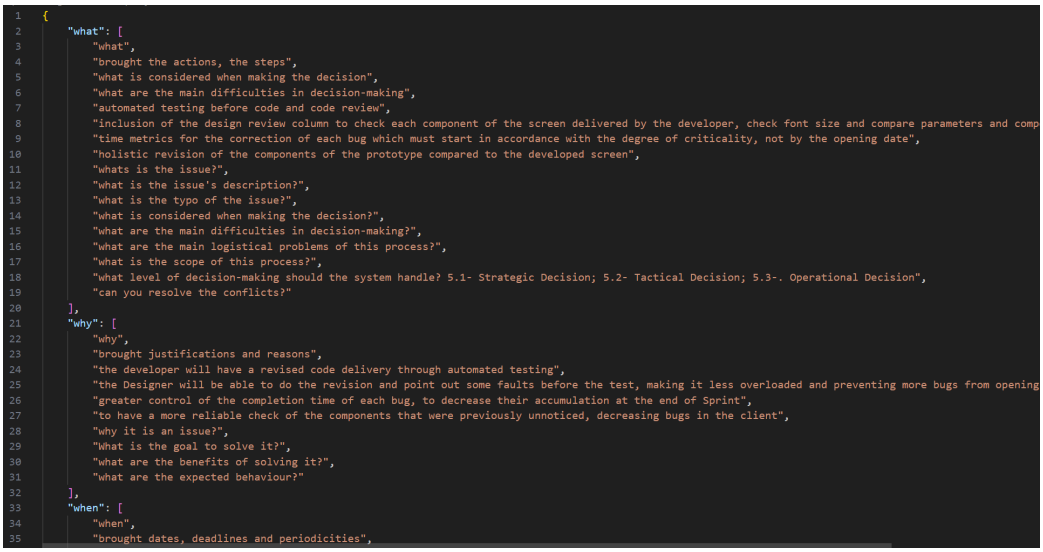

Figure 6.1: Arquivo Examples.json

2) No arquivo gzip_classification atualizar os atributos necessários na classe SelectedDatapara salvar os dados do MongoDB.

```
#creating a class for the the data extraction
class SelectedData:
    def __init__ (self, user, commentsData, issueNumber):
        self.user = user
        self.commentsData = commentsData
        self.issueNumber = issueNumber
```

Figure 6.2: Classe SelectedData do arquivo gzip_classification

3) Também atualizar os atributos da classe DBData, bem com as suas funções. 4) Editar os nomes da database e da collection. 5) Salvar os dados extraídos do MongoDB em uma lista da classe SelectedData, ja fazendo o primeiro filtro das mensagens. 6) Salvar os dados da lista da classe SelectedData para uma lista da classe DBData, nesse momento também ja pode-se fazer o segundo filtro. 7) Após ter as classes prontas, a conexão com o banco de dados estabelecida, os dados selecionados, pode-se

6.7 Considerações Finais

O manual de uso relatou as funcionalidades, potenciais usuários, benefícios, e uso do sistema. Vale resaltar que o sistema ainda está em evolução para a tender possíveis novas necessidades da pesquisa realizada pela autora e seu orentador.

```python
class BDData:
    def __init__ (self, user, commentsData, issueNumber):
        self.user = user
        self.commentsData = commentsData
        self.issueNumber = issueNumber
        self.firstResultData = ""
        self.firstResultProbability = 0
        self.secondResultData = ""
        self.secondResultProbability = 0

    def resultK1(self, firstResultData, firstResultProbability):
        self.firstResultData = firstResultData
        self.firstResultProbability = firstResultProbability

    def resultK2(self, secondResultData, secondResultProbability):
        self.secondResultData = secondResultData
        self.secondResultProbability = secondResultProbability

    def printResults (self):
        print(f"\nUser: {self.user}")
        print(f"\nComments: {self.commentsData}")
        print(f"\nIssue Number: {self.issueNumber}")
        print(f"\nfirst Result Data: {self.firstResultData}")
        print(f"\nprob first Result Data: {self.firstResultProbability}")
        print(f"\nsecond Result Data: {self.secondResultData}")
        print(f"\nprob second Result Data: {self.secondResultProbability}")
        print(f"\n\n******************\n\n")
```

Figure 6.3: Classe Selected Data do arquivo gzip

7
Cenários de uso do sistema

A seguir serão apresentados dois cenários em que o usuário teria sucesso ao uso do sistem e dois cenários que ele teria problemas de uso desse sistema.

7.1
Cenários de utilização do sistema com sucesso

7.1.1
Primeiro Cenário

Usuário: Pesquisador da área de engenharia de software, tem familiaridade com linguagem de programação e deseja fazer pesquisa em relação ao desenvilviemnto de software aberto no GitHub.

Contexto: O pesquisador irá fazer classificações nos textos dos comentários de issues do GitHub para seu trabalho de pesquisa.

Narrativa: Considerando que o sistema está disponível no GitHub, o pesquisador poderá fazer o download dele, adaptar dos acessos ao mongoDB e as classes para as que serão utilizada por ele e realizar o trabalho de classificação e salvar os resultados.

7.1.2
Segundo Cenário

Usuário: Gestor (product manager)

Contexto: O Gestor quer melhorar o desempenho e a comunicação de sua equipe então precisa entender quais são as maiores dificulades dos desenvolvedores no desenvolvimento do produto.

Narrativa: O Gestor poderá fazer o download do código via GitHub, fazer a classificação das perguntas realizadas nos projetos de sua equipe para compreender as maiores dificuldades de sua equipe.

7.2
Cenários que o usuário tem problemas para utilizar o sistema.

7.2.1
Primeiro Cenário

O sistema fornecido considera que o usuário tenha conhecimento na linguagem python para fazer as modificações necessárias para reutilização. Então o primeiro cenário em que o usuário teria problemas em utilizar o sistema é o que ele não tenha esse conhecimento. O sistemas não possui interface.

7.2.2
Segundo Cenário

O segundo cenário é o que o usuário não tenha os os comentários e metricas do GitHub, neste caso o usuário não teria como alimentar o sistema.

8
Considerações finais

Este documento apresentou o sistema que foi desenvolvido para apoiar a pesquisa da autora no trabalho de conclusão de mestrado. O sistema ainda passará por melhorias conforme necessidade da pesquisa.

9
Bibliography

BARBOSA, C. et al. Beyond the code: Investigating the effects of pull request conversations on design decay. 2023. Cited in page 2.

HIPPEL, E. V. Open source software projects as user innovation networks. **Perspectives on free and open source software**, MIT Press Cambridge, MA, p. 267–278, 2005. Cited in page 1.

JIANG, Z. et al. Less is more: Parameter-free text classification with gzip. **arXiv preprint arXiv:2212.09410**, 2022. Cited 4 times in pages 3, 4, 13, and 16.

OHNO, T. **Toyota production system: beyond large-scale production**. [S.l.]: Productivity press, 2019. Cited in page 2.

SALVADORI, B. G.; MAGNAGO, P. F.; DUTRA, A. C. Project based on agile methodologies by dmaic. In: **ICEIS (2)**. [S.l.: s.n.], 2021. p. 337–344. Cited 4 times in pages 2, 4, 13, and 15.

SANTOS, F. et al. Tag that issue: Applying api-domain labels in issue tracking systems. **arXiv preprint arXiv:2304.02877**, 2023. Cited 4 times in pages 2, 4, 13, and 16.

SANTOS, G. N. P.; LUCENA, C. J. P. D. Agile meeting method for building intelligent decision support systems. **MethodsX**, Elsevier, v. 11, p. 102311, 2023. Cited 4 times in pages 2, 4, 13, and 15.